HETED-HÉT

commodore 64

1·2·3

ELŐSZÓ A MÁSODIK KIADÁSHOZ

Mindannyian, akik részt vettünk a Hetedhét Commodore 64 című könyv létrehozásában, nagy örömmel vettük tudomásul, hogy munkánkat a közönség is, a kritika is kedvező fogadtatásban részesítette. Olyannyira, hogy hamar elő kellett rukkolnunk a második kiadással, mert az előző alig pár hét alatt mind egy szálig elfogyott.

Olvasóink kérésére - és a nyomdai átfutási idő csökkentése érdekében - az eddigi három füzetet egy kötetben foglaltuk össze, s ahol észrevettük, javítottuk az előző kiadás hibáit is. Az új kiadás árát így alacsonyabban szabhattuk meg, mint az előzőé volt. Reméljük, olvasóink emiatt nem vonják meg rokonszenvüket a könyvtől...

Budapest, 1985. szeptember

Novotrade RT.

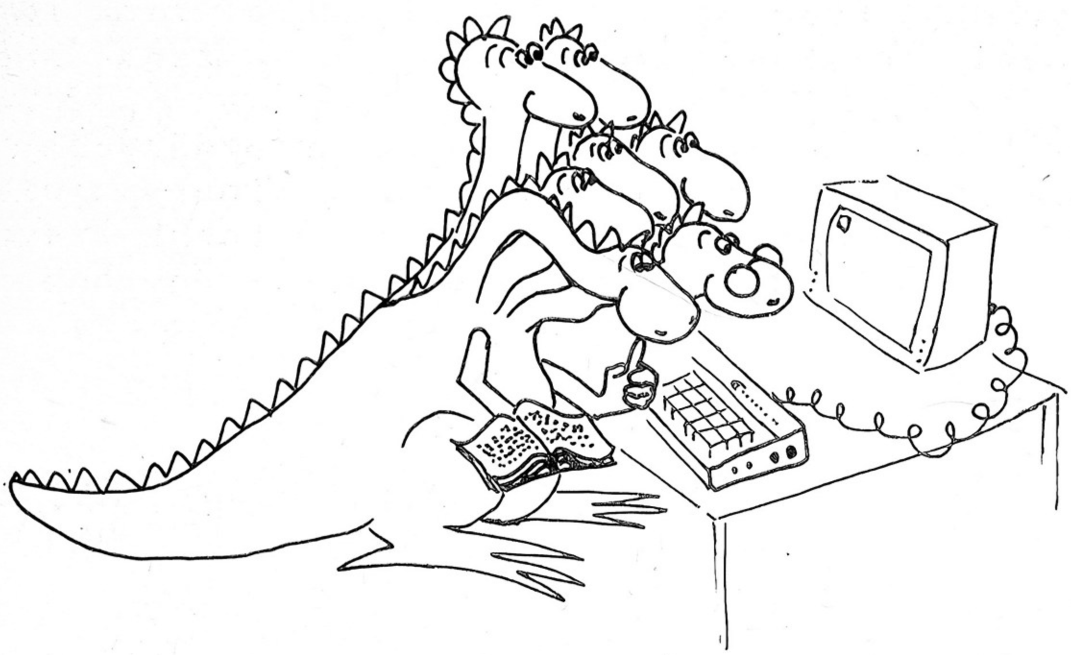
HETEDHÉT

Commodore 64

MÁSODIK HÉT

EZ MÁR A MÁSODIK KÖTET

Keresztülrágtuk magunkat az első köteten, és talán nem is eredménytelenül. Van már fogalmunk arról, hogy a Commodore 64 valóban sokoldalú és jó játékszer, ha megfelelően tudunk bánni vele. Ez a kötet arra szolgál, hogy az eddigi ismereteink mellé újabbakkal szolgáljon. A számítógép egyre engedelmesebb lesz, ahogy egyre jobban beszéljük a nyelvét, és épp ezáltal egyre több örömöt szerez.



Beszéltünk már a változókról, amikor a FOR...NEXT ciklusról volt szó.

Létezik még kétfajta változó, úgyhogy ideje rendet teremteni köztük. (Ez most matek lesz, de nem nagyon veszélyes.)

Kezdjük egy kicsi programmal.

```
10 LET A=5
20 ? A
RUN
```

(A 10-es sor, ha angol nyelvű mondatnak értelmezzük, azt jelenti: LEGYEN A 5-tel egyenlő. Ez a LET a Commodore BASICben elhagyható, de mi az áttekinthetőség kedvéert igyekszünk rendszeresen használni.)

A gép kiírja az 5-ös számot, és azt mondja: READY.

Itt készítettünk egy változót: A lett a neve, és mindjárt értéket is adtunk neki: 5. Ha ezt a programot most tovább írnánk, az A betű mindig ötöt jelentene, egészen addig, amíg más értéket nem adunk neki.

Sok értelme ennek így önmagában nincs, hiszen az A betűt leírni ugyanakkora fáradság, mint az 5-ös számjegyet. Tudni kell azonban, hogy nem muszáj nekünk előre megadnunk a változónk értékét; rábízhatjuk ezt a gépre.

```
10 LET A=5
20 LET B=4
30 LET C=A+B+76
40 PRINT C
```

RUN után a program kiírja a 85-öt. Ez még mindig nem több, mint amit egy átlagos nyolcéves gyerek egy zseb-számológéppel meg tud csinálni, de csak nyugalom, ez csupán az alap, ennél sokkal komplikáltabb is lesz még.

```
10 LET A$="AGOL"
20 LET B$="B"
30 LET C$="Y"
40 LET D$=B$+A$+C$
50 PRINT D$
```

Ezeket a változókat szöveges (string) változóknak nevezzük. Programunkból kiderül, hogy a szöveges változókat ugyanúgy összekapcsolhatjuk + jellel, mint a - nevezzük nevén - valós változókat.

Kiderül továbbá az is, hogy a szöveges változók elnevezésében szerepelnie kell a $ (dollár) jelnek, hogy megkülönböztessük a többi változófajtától. Próbáljuk meg kitörölni a $ jelet a program 30-as sorából, és futtassuk le: TYPE MISMATCH ERROR IN 30 - fogja üzenni a gép, ami azt jelenti, hogy nem olyan karaktert kapott, amilyet várt. Ha nincs ott a $ jel, akkor a gép számjegyet vár az egyenlőségjel után, és helyette betűt, azaz szöveges változót kapott.

Természetesen nemcsak betűből állhat a szöveges változó, hanem grafikus jelekből, írásjelekből (kivéve az idézőjelet), meg szóközökből és számjegyekből is, de akkor is idézőjelbe foglaljuk, tehát ilyenkor is kell a $ jel.

```
10 LET A$="19"
20 LET B$="85"
30 PRINT A$;B$
```

Ekkor a gép semmiféle műveletet nem végez a számjegyekkel, hanem egyszerűen kiírja őket - a pontosvessző miatt szorosan egymás mellé -: 1985. (A pontosvessző helyére írhattunk volna + jelet is, vagy éppen semmit, az eredmény ugyanaz lenne.)

A változókat elnevezhetjük sokféleképpen, de azért vigyázni kell bizonyos dolgokra. Például ezt a három változónevet nem használhatjuk akármire: TI, TI$ és ST. Ezeket a gép különleges célokra tartogatja - némelyikről később még lesz szó.

De van még más is, amire ügyelnünk kell. Például:

```
10 LET KABAT$="KABAT"
20 LET KALAP$="KALAP"
30 PRINT KABAT$,KALAP$
```

Futtassuk le a programot. A gép megbokrosodott! Ezt írja ki:

```
KALAP     KALAP
```

Persze nem a gép a vétkes: mi követtünk el hibát – de hát nem tudtunk egy fontos szabályt: azt, hogy adhatunk jó hosszú neveket is a változóinknak, a számítógép azonban csak az első két karaktert veszi figyelembe, a többit nem. A 10-es sorból csak ennyit érzékel: KA$="KABAT". A 20-asból meg ezt tudja meg: KA$="KALAP". És – mint a katonaságnál – itt is az utolsó parancs az érvényes, ezért a 20-as sort olvasva a gép kicseréli a 10-es sor információját. Vigyázzunk tehát a változók elnevezésénél arra, hogy ugyanazzal a két betűvel ne kezdjünk nevet, mert a későbbi felülírja az előbbit!

```
10 LET MARCIPAN$="MARCIPAN"
20 LET MAZSOLA$="MAZSOLA"
```

Ez így nem jó. Úgy viszont már jó, ha azt mondjuk:

```
10 LET M1ARCIPAN$="MARCIPAN"
20 LET M2AZSOLA$="MAZSOLA"
```

A gép ugyanis ebben az esetben csak az M1-et és az M2-t figyeli, a többivel nem törődik.

Bizonyítsuk be.

```
10 LET M1ARCIPAN$="MARCIPAN"
20 LET M2AZSOLA$="MAZSOLA"
30 PRINT M2$,M1$,M1$,M2$
RUN
```

```
MAZSOLA   MARCIPAN  MARCIPAN  MAZSOLA
```

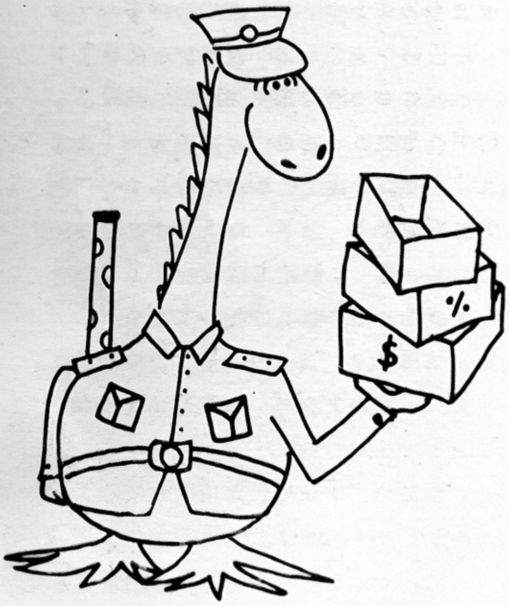
Összefut az ember szájában a nyál.

És ezen kívül is kell még egy dologra vigyázni. Arra, hogy a változók elnevezésében az első karakter mindig betű legyen, különben SYNTAX ERRORt üzen a gép.

A harmadik változótípus az egész változó. Ezt az különbözteti meg a valós változótól, hogy csak egész szám lehet az értéke, törtszám nem. Ezt a típust ritkán használják a BASIC programokban, helyette is inkább a valós változó szerepel. Mint a stringnek, ennek is van egy megkülönböztető jele. a %

Tegyük egymás mellé a három típust, hogy tudjuk, melyik milyen.

TÍPUS	JELE	PÉLDA
Valós	nincs	7.84
Egész	%	103
String	$	"KALAP"



(Emlékezzünk csak vissza: a BASIC nyelvben a tizedestörtekben nem vesszőt használunk, hanem pontot!)

Tudnivaló még: egyazon programon belül különböző típusú változókat elnevezhetünk ugyanúgy - az ABC$ és az ABC két külön fogalom a számítógép számára; akármilyen hosszú nevet adunk a változónknak, a jel eligazítja a gépet, ebből nem lesz kavarodás. Legföljebb mi kavarodunk bele, ha hosszabb, sokváltozós programot írunk.

Összefoglalásul: a változókat képzeljük úgy, mint egy-egy üres dobozt, amelyet mi töltünk meg tartalommal a programozás során, és a tartalmat bármikor ki is cserélhetjük.

Sőt, most megismerkedünk egy új fogással, hogy a program futása közben is csereberélhetjük a dobozok tartalmát. Ehhez egy új BASIC utasítást kell megtanulnunk, éspedig ezt: INPUT. Megint csak angol szó, azt jelenti: beadás, bevitel. Ha a számítógép ilyen utasítással találkozik, akkor megállítja a program futását, egy kérdőjelet rajzol a képernyőre, és várja, hogy mi beírjunk valamit. A program csak akkor indul tovább, ha megnyomjuk a RETURNt.

```
10 INPUT A$
20 LET B$="LOM"
30 LET C$=A$+B$
40 PRINT C$
```

Ez a program azt próbálja ki, hogy hány "-LOM" végű szót ismerünk. Futtassuk le!

A RUN után a képernyő bal szélén megjelenik egy kérdőjel. Gépeljük be: IRODA, és nyomjuk meg a RETURNt.

A gép kiírja: IRODALOM.

Ha újra futtatjuk, és a kérdőjel után azt írjuk: ALKA, az ALKALOM szó alakul ki a RETURN megnyomása után.

Több INPUT utasítást is írhatunk ugyanabba a programba. Javítsuk ki az előzőt így:

```
10 INPUT A$
15 INPUT Q$
20 LET B$="LOM"
30 LET C$=A$+B$+Q$
40 PRINT C$
```

A RUN után megjelenik a kérdőjel, írjuk be: AKAC. A RETURN megnyomása után egy újabb kérdőjel tűnik fel az előző alatt. Ide írjuk azt: BOS. A RETURN után a gép felelete: AKACLOMBOS.

Nézzük át tehát, milyen neveket adhatunk változóinknak, mik ennek a szabályai.

PRINT = 193 (Nem jó név, mert a PRINT egy létező BASIC utasítás, és az ilyenek foglaltak)

H1 = 710 (Jó név, mert az első karaktere betű)

1H = 07 (Nem jó, mert az első karakter nem betű)

FORM = 198 (Nem jó, mert benne van egy BASIC utasítás, a FOR)

PR = 204 (Jó, mert a PRINT is így kezdődik ugyan, de a változónévben nincs benne az egész BASIC szó)

TI$ = "HUSZONKILENC" (Nem jó, mert foglalt; egyike a három névnek, amit nem használhatunk.)

SNRDLUYWIMMXNCHOCH = 1 (Jó név, csak elég nagy hülyeség)

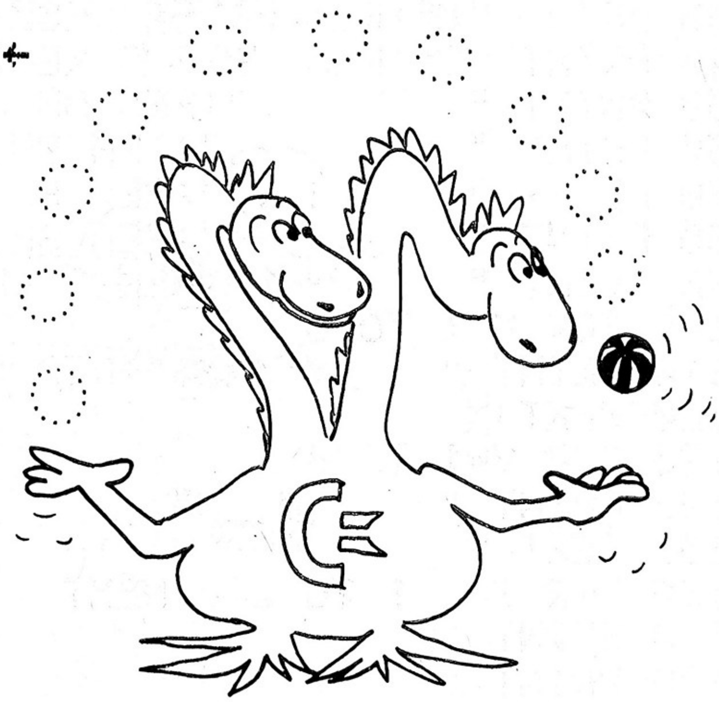
Írjuk ki a képernyőre a svéd Siv Widerberg Szerelem című versét – stringek használatával!

```
5 PRINT"□" : REM SHIFT+CLR/HOME
10 POKE53280,2:POKE53281,1
20 PRINT"▪" : REM CTRL+3
30 S$="STEIN-MALTENAK "
40 N$="NAGY, VOROS ES ELALLO"
50 F$="FUL"
60 E$="E VAN."
65 A$="A "
70 SZ$="NEKEM TETSZIK"
80 PRINTS$;N$
90 PRINTF$;E$
100 PRINTSZ$
110 PRINTA$;N$
120 PRINTF$
```

Ha lefuttatjuk a programot, elolvashatjuk a listából eléggé rejtélyesnek tűnő, kedves kis verset. Közben pedig, ha összehasonlítjuk a programlistát a kész verssel, a stringekről tanultakat is átismételhetjük.

Emlékszünk rá, hogy a hatodik napot egy labdát mozgató programmal fejeztük be. Eléggé siralmas kis programocska volt ez, ami azt illeti: zötyögve, villogva haladt a golyó, és ráadásul csak balról jobbra. Most már tudunk eleget ahhoz, hogy megírjuk ezt a programot sokkal rövidebben, és a golyó nemcsak jobbra fog haladni, hanem ide-oda pattog a képernyőn, amíg a STOP gombbal meg nem állítjuk.

```
10 REM JOBBRA HALAD
20 PRINT "♥"
30 FOR JO=1 TO 39
40 PRINT " ●▌";:REM CRSR←
50 FOR K=1 TO 10:NEXT
60 NEXT JO
100 REM BALRA HALAD
110 FOR BA=1 TO 39
120 PRINT "▌● ▌▌";
130 FOR K=1 TO 10:NEXT
140 NEXT BA
150 GOTO 30
```



Néhány megjegyzést már bele is írtunk a programba (REM-mel elkülönítve), de még tartozunk egy magyarázattal: mitől mozog a golyó?

Nézzük meg a stringünket! A kötelező idézőjel után egy üres szóközt hagytunk. Ez az üres hely fog kirajzolódni a golyó előtti oszlopban – és ezáltal kitörli az előző golyót – így fog majd úgy tűnni, hogy a golyó átkerült az egyik oszlopból a másikba. A string harmadik eleme eggyel balra lépteti a cursort. Enélkül az üres szóköz a string elején nem az előző golyót törölné ki, hanem az utána következő üres helyre kerülne, és így a golyó nem haladna előre, hanem egyre több golyó rajzolódna ki.

Ugyanez a helyzet a balra haladó golyóval: ott az üres szóköz a golyó jobb oldalán foglal helyet, hiszen az új golyó mindig az előzőtől balra rajzolódik ki; a cursort pedig háromszor kell balra mozgatni a string végén, hogy a következő lépésben egy hellyel az előzőtől balra kezdje a kiírást.

A stringekkel most már eléggé jól tudunk bánni ahhoz, hogy a tornázó robot programját – ld. a hetedik napot az előző kötetben – meg tudjuk írni ebben a formában is. Ugye észrevettük, hogy stringnek nevezzük akkor is, ha nem értelmes szövegből áll, hanem grafikus jelekből?

```
10 POKE 53280,6:REM KEK LESZ A KERET
20 POKE 53281,6:REM KEK LESZ A HATTER
30 PRINT "■":REM FEHER LESZ A ROBOT
40 PRINT "♥":REM URES A KEPERNYO
50 PRINT "         ╲╱":REM SHIFT+M,SHIFT+N
60 PRINT "        ╭──╮":REM SHIFT+U,SHIFT+*,SHIFT+I
70 PRINT "        |..|":REM SHIFT+-
80 PRINT "        ╰──╯":REM SHIFT+J,SHIFT+*,SHIFT+K
90 PRINT "          ▌ ":REM C=+L,C=+J
100 FOR X=1 TO 5
110 PRINT "       *▒▒▒▒*":REM C=++
120 NEXT X
130 FOR Y=1 TO 4
140 PRINT "       *  *"
150 NEXT Y
160 FOR I = 1 TO 200:NEXT
170 PRINT "♥"
180 PRINT "       ╲╱"
190 PRINT "      ╭──╮"
200 PRINT "      |..|"
210 PRINT "      ╰──╯"
220 PRINT "        ▌"
230 PRINT "*****▒▒▒▒*****"
240 FOR Z=1 TO 4
250 PRINT "      ▒▒▒▒"
260 NEXT Z
270 PRINT "      *  *"
280 PRINT "     *    *"
290 PRINT "    *      *"
300 PRINT "   *        *"
310 FOR I=1 TO 200:NEXT
320 GOTO 40
```

Ezt a programot is, csakúgy, mint az előzőt, a STOP gombbal lehet leállítani. Utána el lehet szórakozni azzal, hogy átalakítjuk, ahogy szoktuk; egyetlen számjegy átjavításával nyurga vagy tömzsi robotot csinálhatunk stb.

Most még egy kicsi program, sokféle változóval, INPUTtal, hogy világosan értsük, miről van szó. Gépeljük be ezt a kis programot, amely először megkérdezi a személyi adatainkat, aztán a képernyőre kiírja a névjegyünket.

```
0 REM NEVJEGY
10 PRINT"♥"
20 PRINT "MELYIK VAROSBAN ELSZ?"
30 INPUT V$
40 PRINT"MELYIK UTCABAN?"
50 INPUT U$
60 PRINT"MI A HAZSZAM?"
70 INPUT H
80 PRINT"MI AZ IRANYITOSZAM?"
90 INPUT I%
100 PRINT"♥"
110 PRINT"IME A LAKCIMED:"
120 PRINT,I%;V$
130 PRINT,U$;" UTCA";H
```

Figyeljük meg a változókat. A város neve (V$) szöveges változó, akárcsak az utcanév (U$). Az irányítószám egész változó, mert ott törtszámnak nincs értelme. A házszám viszont valós változó, mivel ha valaki például az Avokado utca 24/26-ban lakik, az ilyen formában beírhatja: 24.26 - ezt ugyan tizedes törtnek fogja fel a gép, de mivel számolnia nem kell vele, csak kiírnia, ezért nem okozunk bajt.

Egészítsük ki a programot úgy, hogy telefonszámot is kérdezzen, és a kiiráskor a lakcím alá tegye!

Épp eleget tudunk a változókról ahhoz, hogy megírjunk egy olyan programot, amely egy táborozó gyerek levelét írja ki a képernyőre. A levél szövege az alábbiakban olvasható, a programban a kipontozott részek helyére különféle fajtájú változókat írjunk, méghozzá úgy, hogy az értékük INPUT formájában kerüljön a helyére.

"Kedves ...!
Kérlek, küldj sürgősen ... darab ...-t, mert elfogyott a ...-m, és ha ...-ra sem lesz, akkor ... "

Ezt a programot most segítség nélkül írjuk meg - könnyítésül annyit, hogy a kipontozott részek helyére kerülő változókat mi így neveztük el: CIMZETT$, DARAB, MI$, MAGYARAZAT$, HATARIDO$, KOVETKEZMENY$; azaz, röviden: CI$, DA, MI$, MA$, HA$, Ko$

Sok mindent tudunk már a számítógép képességeiről, de egy igen fontos lehetőségről még szót sem ejtettünk.

Eddig kétféle módon dolgoztattuk a gépet: vagy úgy, hogy sorra vette a programsorokat, a legalacsonyabb sorszámúval kezdve, vagy pedig - mint mondtuk - "külön kérésre" kilépett ebből a folyamatból, és a GOTO utasítás hatására átugrott az általunk előre megnevezett programsorra.

A gépet azonban feltételes módban is utasíthatjuk. Hogy ez hogyan megy, arra nézzünk egy nagyon rövidke kis programot.

```
10 LET A=A+1
20 PRINT A;
30 IF A<100 THEN GOTO 10
40 END
```

No, itt aztán van mit nézni. Először is itt a 10-es sor, amely ellentmond mindennek, amit eddig matematikából tanultunk. Hogy lehet A egyenlő (A+1)-gyel? Hát úgy, hogy ez nem matematika, hanem BASIC, és ez nem egyenlet, hanem a gépnek adott utasítás: azt jelenti pontosan: "az A-nak nevezett változó értékét növeld meg eggyel".

Ha külön nem mondunk mást, akkor a gép az A első értékét 0-nak veszi, és onnan növeli eggyel minden alkalommal, amikor erre a programsorra ér.

A 20-as sor, ezt régóta tudjuk, kiírja az A mindenkori értékét, és utána nem lép rögtön új sorra, mert ott a pontosvessző.

Az igazi érdekesség a 30-as sorban van. Az a talán ismeretlen jel azt jelenti: "kisebb, mint". (Figyelem! a fejezet végén összegyűjtve bemutatjuk ezeket a jeleket - aki türelmetlen, már most odalapozhat!) És ott van még az a két ismeretlen szó:

IF és THEN. Ezeknek van pontos magyar megfelelőjük: HA és AKKOR. E két szó az angolból változatlan értelemben került át a BASICbe, vagyis egy feltételes módot vezet be a programba. BASICből magyarra fordítva a sort, ilyen mondatot kapunk tehát: "Ha A kisebb, mint száz, akkor menj a 10-es sorra!"

A gép tehát egyesével növelgeti A értékét, és minden alkalommal ellenőrzi, hogy elérte-e a 100-at. Ha A értéke 100-zal egyenlő, akkor a 30-as sor betöltötte szerepét, a program tovább fut: rá a 40-es sorra, és ott azt találja: END.

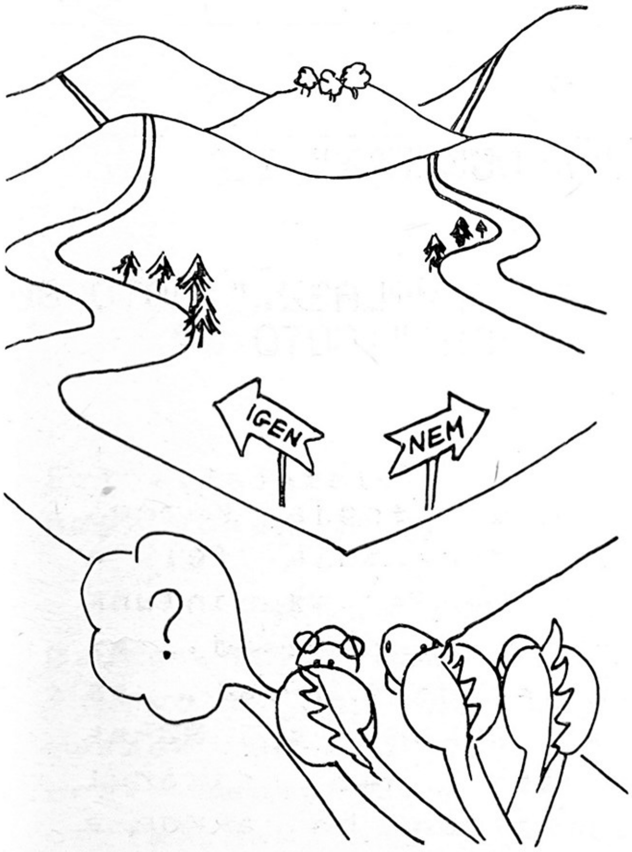
Ez új utasítás ugyan, de jól ismeri mindenki, aki látott már angol filmet; a szó azt jelenti: VÉGE. Ha tehát az A értéke eléri a százat, akkor a program futásának vége.

Világosabban fogjuk látni az IF...THEN szerkezet lényegét, ha a programunkat átalakítjuk.

```
10 LET A=A+1
20 PRINT A;
30 IF A=100 THEN END
40 GOTO 10
```

Az előbbi esetben így okoskodott a gép, amikor a 30-as sorra ért: "Kisebb az A értéke, mint 100? Igen. Akkor vissza kell mennem a 10-es sorra." És vissza is ment mindaddig, amíg a feltétel nem teljesült, vagyis ameddig A valóban kisebb volt 100-nál. Ha A elérte a 100-at, vagyis nem volt kisebb, akkor továbblépett a programban.

Most, a második esetben ugyanolyan feltételes módú az utasításunk, de a gép gondolatmenete más. "Egyenlő A értéke 100-zal? Nem? Akkor megyek tovább." És ment tovább a következő sorra, anélkül, hogy megvizsgálta volna, mi áll a THEN után a sorban. A THEN utáni utasítást ugyanis a számítógép csak akkor hajtja végre, ha az IF utáni feltétel igaz. A THEN egy zárt ajtóhoz hasonlítható, amely csak akkor nyílik ki, ha teljesül a feltétel, amelyet az IF szabott.



(Úgy szokták magyarázni, hogy az IF...THEN szerkezet két ágra nyílik szét: az "igen" ágra és a "nem" ágra. "Igen" ágnak nevezik azt, amelyik a THEN utáni utasításokból áll. Ide akkor jut el a program, ha az IFben megszabott feltételek teljesülnek. Ha nem így történik, akkor a "nem" ágra fut a program, ez az ág pedig az IFet követő első sor első utasításával kezdődik.)

Második programváltozatunknak van egy másik tanulsága is: nem biztos, hogy a program vége egybeesik a programlista végével. Ha az IF...THEN szerkezet úgy kívánja, az END utasítás után ezer programsor is következhet még.

Csinaljunk egy apró módosítást a program második változatában.

```
10 LET A=A+3
20 PRINT A;
30 IF A=100 THEN END
40 GOTO 10
```

Futtassuk le! Meglepetés fog érni: a program jócskán túlfut a százon, és úgy írja tovább a képernyőre a számokat, mintha soha nem akarná abbahagyni. Le kell állítanunk a STOPpal.

Mi történt itt? Egyszerű: ha hármasával számolunk, akkor az IFben megszabott feltétel sohasem teljesül: a 100 nem osztható hárommal. A THEN kapuja csukva marad, az ENDre nem fut rá a program.

A megoldás: egy uj jelet kell használni, azt, hogy >=. (Nagyobb, vagy egyenlő.) a 30-as sor tehát így alakitandó át:

```
30 IF A>=100 THEN END
```

Ez már egyértelmű: az A értéke most már legfeljebb kettovel haladhatja tul a szazat.

Az IF...THEN sok-sok BASIC programban jön majd a segitsegunkre. Peldaul a következőben, amely kiprobalja, hogy milyen fejszamolo, aki használja. (Persze csak alapfokon.)

```
0 REM FEJSZAMOLTATO
10 PRINT "HELLO!"
20 FOR I=12 TO 60 STEP 3
30 PRINT "USS BE 2 SZAMOT, AMELYEK OSSZEGE";I
40 INPUT A
50 INPUT B
60 IF A+B=I THEN PRINT "OKE, HELYES A VALASZ.":GOTO 80
70 PRINT "EZ NEM STIMMEL, PROBALD UJRA!":GOTO 30
80 NEXT I
```

Az egész programot egy FOR...NEXT ciklus fogja közre, amely 12 és 60 között hármasával lépkedve adja fel a számokat. A két INPUT sorban adjuk be a szerintünk helyes két megfejtést, amit a program összead, az eredményt összehasonlítja az általa feladott I-vel, és ha a kettő egyezik ("igen" ág), akkor a 80-as programsorra küld. Ha az összeadást nem sikerül megoldanunk, tehát rossz számokat gépeltünk be, akkor a THEN után következő "igen" ág helyett a program a "nem" ágra ugrik, enyhén megró a pontatlan feleletért, és - újból feladja az előző számot! (Világos, hiszen nem lépett tovább a NEXT I-t tartalmazó 80-as sorra, hanem előbb visszafordult.)

Vállaljunk egy kis kockázatot: egy fontos tanulság kedvéért rontsuk el a jó programunkat! Töröljük ki a 60-as sorból a GOTO 80 részt! A program elveszíti a józan ítélőképességét: ha helyes választ adunk, akkor előbb megdícsér, de utána ugyanúgy megró, mintha hibáztunk volna, és ugyanazt a szamot adja fel újra és újra, a végtelenségig.

Vigyáznunk kell erre: a programnak mindig meg kell mondani, hogy mit tegyen, ha a feltételek teljesülnek (tehát az "igen" ág irányát), különben a "nem" ágon folytatja, és elvisz bennünket az erdőbe.

Végül váltsuk be ígéretünket: gyűjtsük ki a két mennyiség egymás közti viszonyát kifejező jeleket!

JEL	JELENTÉS
=	(egyenlő)
>	(nagyobb, mint)
<	(kisebb, mint)
<>	(nem egyenlő)
>=	(nagyobb vagy egyenlő)
<=	(kisebb vagy egyenlő)

Ezt a fejezetet egy kis történettel kezdjük; lehet, hogy sokan ismerik, de talán többen vannak, akik nem.

A későbbi nagy matematikus, Karl Friedrich Gauss (1777-1855) már iskolásgyerekkorában igen jóeszű fiú volt. Egy számtanórán a tanár, akinek valami sürgős maszek munkája akadt, szerette volna, ha nem kell az órát megtartania, ezért egy feladatot adott az osztálynak, ami - gondolta naivan - majd lefoglalja a nebulókat, s ő nyugodtan végezheti a munkáját. "Gyerekek, adjátok össze az egész számokat 1-től 100-ig!" - mondta, és leült dolgozni. Azzal azonban nem számolt, hogy zseni is van az osztályban. A kis Gauss alig pár perc után jelentkezett az eredménnyel: 5050.

A tanár dühösen, de elképedve kérdezte, hogyan számította ki ilyen hamar. Gauss elmagyarázta a módszerét: észrevette, hogy ha az 1-től 100-ig tartó számsor két végén álló számot összeadja, 101-et kap eredményül. Ugyanígy 101 jön ki akkor is, ha a második és az utolsó előtti számot adja össze (2+99), és így tovább, míg a sorozat közepén az 50 és az 51 összege is éppen 101. Ezért a 101-et megszorozta 50-nel, és máris kezében volt a megoldás: 5050.

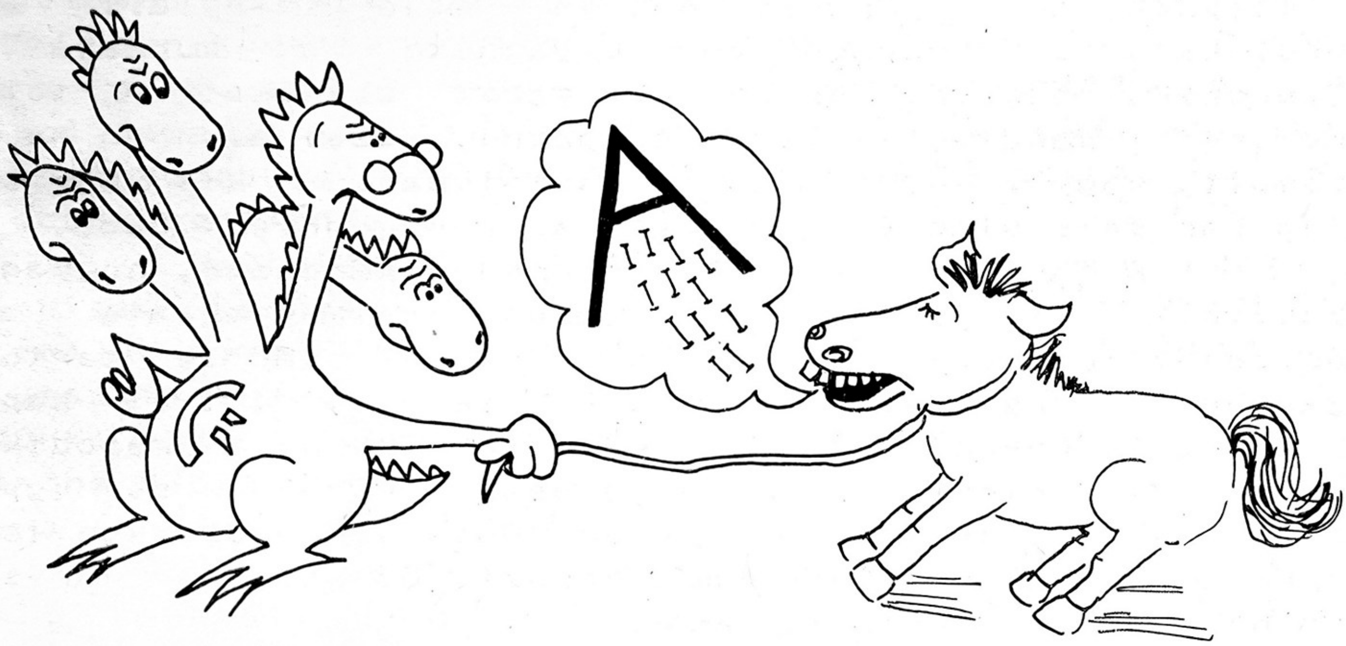
Gauss gyorsan kiszámította az eredményt, de még ő sem versenyezhet a Commodore 64-gyel, ha megfelelő programot írunk rá. Írtunk is: gépeljük be, futtassuk le, aztán beszélünk róla.

```
10 REM GAUSS
20 A=0
30 FOR I=1TO100
40 A=A+I
50 NEXT I
60 PRINT A
70 A=0:I=1
```

```
80 A=A+I:I=I+1
90 IF I>100 THEN PRINT A:END
100 GOTO80
```

RUN után alig egy másodperc telik el, és a gép már írja az eredményt: 5050. De mi van itt? Még egyszer kiírja, az előző alá. Csak nem kétszer számolta ki ez az egy program?

De bizony, pontosan ez történt. Mégpedig azért, hogy szemléletesen be tudjuk mutatni: a FOR...NEXT ciklus és az IF...THEN szerkezet tökéletesen tudja helyettesíteni egymást bizonyos feladatokban - például ebben. A programunk első része (a 20-as sortól a 60-asig) egy FOR...NEXT ciklus, amely két változót használ: az A-t meg az I-t; az I egyesével lépked a számsoron 1-től 100-ig, az A-ban pedig sorban összeadódnak ezek a számok, és a végeredmény kiíródik a képernyőre is.



A program másik része (a 70-es sortól a 100-asig) ugyanezt végzi el egy IF...THEN szerkezettel. Itt érdemes megfigyelni: az "igen" ágon kiíratjuk A értékét, és vége a programnak, a "nem" ágon pedig visszaküldjük a 80-as sorra, hogy növelje tovább az I-t, egészen az utolsó, a 100-as értékig.

Az END utasítás bárhol elhelyezhető a programban, akár ideiglenesen is - ha például valami nem úgy működik a programunkban, ahogyan elterveztük, és szeretnénk megkeresni a hiba forrását. Ilyenkor

valahová beírhatjuk (célszerű persze, ha külön sorba) az ENDet, és a program csak odáig fut, utána READYvel leáll. Ily módon akár sorról sorra ellenőrizhetjük a programunkat, ha az ENDet átrakjuk új és új helyekre. Az END hatására idő előtt leállt programot folytatni is tudjuk, ha begépeljük a CONT utasítást, és megnyomjuk a RETURNt. (CONT a "continue", azaz "folytasd!" angol szó rövidítése.)

Próbáljuk ki: a Gauss-programba írjuk be a két ciklus határára az ENDet (például 65-ös sorszámmal):

```
10 REM GAUSS
20 A=0
30 FOR I=1TO100
40 A=A+I
50 NEXT
60 PRINT A
65 END
70 A=0:I=1
80 A=A+I:I=I+1
90 IF I>100 THEN PRINT A:END
100 GOTO80
```

Most a program csak egyszer írja ki az 5050-et, és csak akkor fut rá az IF...THEN szerkezetes programrészre, ha megmondjuk neki: CONT.

A "nagyobb", "kisebb" jelekkel végezzünk most el egy meglepő kísérletet! Írjuk be ezt a kis programot!

```
0 REM ABC
10 INPUT "ELSO NEV";A$
20 INPUT "MASODIK NEV";B$
30 IF A$<B$ THEN PRINT A$,B$:GOTO 50
40 PRINT B$,A$
50 END
```

Ez a program ABC-sorrendbe állít két bármilyen nevet (vagy más szót). Szokatlan gondolat szöveges változók esetében a "kisebb" jelet használni, hiszen hogyan "kisebb" az A a B-nél? Hát úgy, hogy a betűknek a gépben kódszámuk van, amely az ABC sorrendjében növekszik. Amikor a gép az INPUT sorokban megkapja a két nevet, e kódszámokat hasonlítja össze, és eszerinti sorrendben írja ki őket. Sőt, azt is tudja, hogy az ANNAMARIA később jön az ABC-ben, mint az ANNA!

Fejezzük be egy feladattal: az összeadást gyakoroltató program mintájára írjunk egy olyat, amelyik felad két számot 1 és 20 között, és a szorzatukat kérdezi a használótól; ha helyes választ kap, akkor tovább lép, ha helytelent, akkor korholó szavak kíséretében még egyszer feladja ugyanazt a két számot.

5. NAP — VEKTOROK, MÁTRIXOK ÉS MÁS RIADALMAK

Ha valaki vektorokat emleget előttünk, ne szaladjunk el messzire - ezek sem veszélyesebb fogalmak, mint amelyekkel eddig megismerkedtünk. A BASICben a vektorok olyan változócsoportok, amelyeket azonos néven tartunk nyilván, de megszámozzuk őket, hogy áttekinthetőbb legyen a programunk.

Még ilyen ostobaságot! Hiszen a változók elnevezésében éppen az a poén, hogy jól különítsük őket el egymastol (CIMZETT$; HATARIDO$ - és így tovább...) Most meg azzal jön elo valaki, hogy nevezzük őket egyformán, attól áttekinthetőbb lesz; ez teljes őrület.

Csak türelem, nézzünk néhány példát.

SZÖVEGES VALTOZO	SZÖVEGES VEKTOR
K$ = "KUTYA"	HA$(1) = "KUTYA"
M$ = "MACSKA"	HA$(2) = "MACSKA"
L$ = "LO"	HA$(3) = "LO"
C$ = "CSIRKE"	HA$(4) = "CSIRKE"

A programunkban mindkét fajtát használhatjuk; akár PRINT K$, akár PRINT HA$(1) áll a programban, a KUTYA ugyanúgy megjelenik a képernyőn.

Természetesen számszerű változókat is elnevezhetünk így:

```
A(1) = 1
A(2) = 2
A(3) = 3
```

es igy tovabb. De ezzel meg nem magyaraztunk meg semmit. "Es most mi van? - kérdezheti akárki. - Miért ne hasznaljam a regi, jol bevalt valtozoimat, minek nekem a vektorok?"

Hát először is: a vektorok segítségével könnyebb nyomon követni, mi történik a programban. (Eddig csak ilyen kis nyúlfarknyi programocskákat írogattunk, de amikor már sok száz vagy ezer sorban kell eligazodni, sokkal áttekinthetőbb, ha az összetartozó változókat vektorokba rendezzük.) Emellett rengeteg programozási időt is megspórolunk. Nézzük meg például az alábbi programot!

```
10 PRINT "□"
20 FOR I=1 TO 10
30 PRINT I;"HAZIALLAT";: INPUT HA$(I)
40 NEXT I
50 FOR N=1 TO 10 : PRINT HA$(N)
60 NEXT N
```

No, tessék, futtassuk le a programot, és utána próbáljuk meg megírni ugyanezt vektorok nélkül! Sok időbe fog kerülni, de csak rajta, próbáljuk meg. H0$-tól H9$-ig nevezgethetjük a változóinkat, amit a fenti programban a számítógép végzett el helyettünk, összetartozó tömbként kezelve változóinkat.

Ha valaki nem sajnálta a fáradságot, és megírta a programot, meglátta, mennyi időt takaríthat meg a vektorokkal. De mielőtt továbbmennenk, nezzuk meg gyorsan, hogyan bánt el a programunk a FOR...NEXT ciklusban a vektorváltozókkal.

1) A FOR...NEXT ciklus sorban behelyettesítette az I értékeit a zárójelek közé, ugyhogy a vektort így göngyölítette fel a gép:

```
FOR I = 1 TO 10
 HA$(1) < itt kezdi a ciklust először
  HA$(2) < masodszor ér a FORhoz
   HA$(3) < harmadszor ér a FORhoz
    HA$(4) stb.
     HA$(5)
      HA$(6)
       HA$(7)
        HA$(8)
         HA$(9)
          HA$(10)
NEXT I
```

2) Minden INPUT stringet, amit begépeltünk, sorban megszámozott vektorváltozóként eltárolt a gép.

3) Az 50-es sorban található PRINT utasítás egy másik FOR...NEXT cikluson belül van, ez a ciklus helyettesítette be a számokat 1-től 10-ig a kiírás alkalmával.

Nem véletlen, hogy éppen 10-et választottunk tömbünk nagyságának. Ekkorát engedélyez ugyanis a gép anélkül, hogy előre a tudomására hoznánk, mekkora lesz a vektorunk. Pontosabban 11-et, mert 0-tól 10-ig automatikusan helyettesíti be a számokat a zárójelbe, de afölött már nem. Ekkor a gép hibaüzenetet küld; hacsak előtte észnél nem voltunk, és nem írtuk meg, mekkora vektorra gondoltunk.

Ezt a műveletet dimenzionálásnak nevezik (a dimenzió kiterjedést jelent). Az erre vonatkozó BASIC szó a DIM, és a 10-nél nagyobb vektorunkat így dimenzionálhatjuk:

```
DIM A(150)
```

ami azt jelenti, hogy az A vektornak legfeljebb 151 eleme lehet. (Az elemek értéke - valós változóról lévén szó, valós szám, tehát egész és tört is lehet.) De ugyanígy dimenzionálhatunk stringváltozókat is:

```
DIM AB$(30)
```

ahol az AB$ nevű string 31 elemű lehet.

Lássunk egy kis programot, amely 150 különböző sugarú kör területét számolja ki.

```
10 PRINT"□"
20 DIM KT(150)
30 FOR R=1 TO 150
40 KT(R)=R*R*π
50 NEXT R
60 FOR R=1 TO 150
70 PRINT R,KT(R)
80 NEXT R
```

Egy megjegyzés a 40-es sorhoz: a pi jelet a RESTORE gomb mellett találjuk meg, SHIFTtel hívható elő. Természetesen az R*R (azaz a sugár négyzete) hatványozás formájában is felírható: R↑2. (A felfelé mutató nyíl a BASICben a hatványozás jele; utána áll a hatványkitevő.) Ugye, mennyi gépelést megspórol így az ember?...

Ez a program kissé nehezen lendül mozgásba, hiszen rengeteg számítást kell elvégeznie, de utána mint a villám, ontani kezdi az adatokat, szépen táblázatba szedve.

Ezek a számok 1-től 150 egységig terjedő sugárhosszúságú körök területének adatai, amelyeket persze alig lehet megfigyelni, mert egy szempillantás alatt átsuhannak a képernyőn. Csakhogy a gép a program lefutása után nem felejtette el őket - bármelyiket meg lehet kérdezni tőle. Például:

```
? KT(88)
```

A RETURN megnyomása után megkapjuk a választ:

```
24328.4935
```

A 88 egység sugarú kör területe tehát ennyi - és a gép e percben 1-től 150-ig mindegyiket tudja.

Ha meg akarjuk spórolni az időt, a program 70-es sorát el is hagyhatjuk. Semmi szükség rá, hogy kiírja a sok adatot, amit nem is tudunk felfogni, olyan gyorsan eltűnnek a szemünk elől. Csináljunk a 70-es sor elején helyet 3 betűnek, írjuk be: REM, és ezzel a számokat kiírató sort kivontuk a forgalomból. (Ezzel persze fölöslegessé vált a 60-as és a 80-as sor is; írjunk ezek elé is REMet!) Ilyenkor lefuttatva a programot, egy ideig semmit sem látunk a képernyőn, aztán is csak a READY jelenik meg. A gép azonban most már tudja a tudnivalókat (és tudni is fogja, amíg NEWt nem mondunk neki); bármelyik KT-értékre rákérdezhetünk, sőt az egyes KT-értékekkel műveleteket is végeztethetünk.

Milyen arányban van egymással például az egy egységnyi sugarú kör területe a két egységnyiével?

```
? KT(2)/KT(1)
```

A gép villámgyorsan adja a választ: 4.

Az egydimenziós vektorok mellett előbb-utóbb szükségünk lesz két- és többdimenziós tömbök kezelésére is. Ebben a könyvben azonban nem megyünk tovább a kétdimenziósnál.

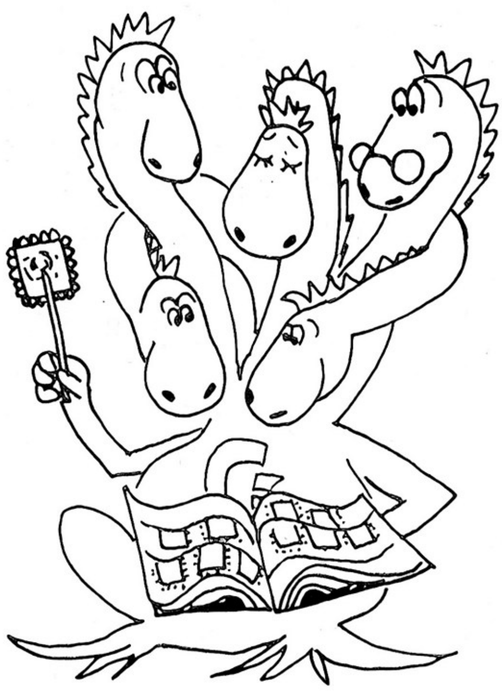
A kétdimenziós tömböt (mátrixot) a legegyszerűbben úgy képzelhetjük el, mint egy táblázatot, amely számozott sorokra és oszlopokra van osztva, valahogy így:

	0.OSZLOP	1.OSZLOP	2.OSZLOP
0.SOR	A(0,0)	A(0,1)	A(0,2)
1.SOR	A(1,0)	A(1,1)	A(1,2)
2.SOR	A(2,0)	A(2,1)	A(2,2)

Ez már egy kissé ijesztőnek is tűnhet, de jusson eszünkbe, hogy még mindig ugyanolyan változókról beszélünk, mint a jó öreg A vagy AB$. Hogy erről megbizonyosodjunk, írjunk be néhány sort, s mindegyik után nyomjuk meg a RETURNt!

```
        XY$(3,1)="KOZONSEGES SZOVEGES VALTOZO  VAGYOK"
: ? XY$(3,1)
        PI(0,0)=3.14159: ? PI(0,0)
        QQ%(7,2)=467 : ? QQ%
```

Lehet, hogy kissé túlmagyarázzuk ezt a részt, de fontos, hogy ha tömbökről, vektorokról, mátrixokról beszélünk, változókra gondoljunk, ne másra. A nap végén egy kis bélyegalbum-rendező programot közlünk; gépeljük be gondosan, és utána futtassuk le. Sok gyakorlati haszna nincs, mert a begépelt adatokat nem őrzi meg – sokkal inkább arra jó, hogy figyelmesen végigolvasva a programlistát, megtudjunk egyet-mást a többdimenziós tömbökről.



```
10 PRINT"♥"
20 PRINT"BELYEGALBUM-RENDEZO"
30 PRINT:PRINT"HANY OLDAL,SOR,OSZLOP VAN?"
35 INPUT"(AZ ADATOK KOZE TEGY VESSZOT!)";OL,SO,OS
40 DIMBE$(OL,SO,OS)
50 INPUT"HANY BELYEGET AKARSZ ELHELYEZNI";N%
60 PRINT:FORI=1TON%
70 INPUT"OLDAL SZ.->";OD
80 INPUT"SOR SZ.->";SR
90 INPUT"OSZLOP SZ.->";OP
100 INPUT"HONNAN VALO A BELYEG:";BG$
110 BE$(OD,SR,OP)=BG$
120 NEXTI
200 REM VISSZAKERDEZES
210 PRINT"♥":INPUT"HANYADIK OLDALT AKAROD MEGNEZNI";OO
220 FORI=1TOSO
230 FORJ=1TOOS
240 IFBE$(OO,I,J)=""THENBE$(OO,I,J)="NINCS"
250 PRINTI;"SOR";J;"OSZLOP: ";BE$(OO,I,J);" BELYEG"
260 FOR K=1TO500:NEXT K
270 NEXTJ
280 NEXTI
290 INPUT"AKARSZ UJRA LISTAZNI? (I/N)";LI$
300 IF LI$="I" THEN GOTO 210
310 INPUT"AKARSZ UJ BELYEGET BETENNI? (I/N)";BE$
320 IF BE$="I" THEN GOTO 50
330 END
```

Ez a rész meglepő tulajdonságait tárja fel a számítógépnek: olyan benyomásunk támad, mintha a gép önállóan döntene bizonyos dolgokban. Erről persze nincs szó; a gép mindig emberi utasításokat hajt végre, most mégis úgy tűnik, mintha az ellenkezője volna igaz.

Kezdjük egy programmal. Ez tulajdonképpen megint fejszámolásra késztet: két, a gép által megadott számjegyet kell fejben összeadnunk és begépelnünk.

```
10 REM GYORSSZAMOLO
20 X=INT(10*RND(1))+1
30 Y=INT(10*RND(1))+1
40 PRINTX"+"Y"=";
50 INPUT Z
60 IF Z=X+Y THEN PRINT "SZUPER!":GOTO20
70 PRINT"HOPPA, EZ NEM JOTT OSSZE. PROBALD UJRA.":GOTO40
```

Indítsuk el a programot, és játsszunk vele egy darabig, hogy érezzük, milyen rendszer szerint adja a számokat. Ha már meguntuk, ebből a programból nem könnyű csak úgy, a STOP megnyomásával kilépni, hiszen a program olyan, hogy csak számjegyek begépelését fogadja el (az INPUT szó után valós változót vár, amint az a string fajtájából kiderül). A program leállításának módja a következő: nyomjuk le a RUN/STOP gombot, és amíg nyomva tartjuk, nyomjuk le a RESTORE gombot is (mostanáig megúsztuk ennek a kezelését). Ez az egy billentyű hangyányit másképpen jár, mint a többi a Commodore 64-en: egy kicsit rá kell koccantani ahhoz, hogy a program leálljon.

Figyelem! Agyoncsapni nem kell! Csak egy kicsit koccantani rajta!

Ebben a a programban két ismeretlen kifejezés van: az INT és a RND.

Az INT az integer, azaz egész szám rövidítése.

A RND egy sokkal érdekesebb dolog a mi számunkra, mert rengeteg játéklehetőséget kínál. A három betű egy angol szó rövidítése: RANDOM, azaz véletlenszerű. Ha ezzel az utasítással találkozik, a számítógép magától helyettesíti be a változó értéket, ahogyan neki éppen jól esik. No persze nem éppen teljesen szabadon, hanem bizonyos korlátok között. A korlátokat is ott látjuk a programban, és nemsokára fel is fogjuk ismerni őket.

A RND véletlenszerű számokat hoz létre, mégpedig 0 és 1 között. Ha tehát egy ilyen programot futtatunk:

```
10 ? RND (1)
20 GOTO 10
```

akkor a képernyőn csupa 1-nél kisebb szám fog fölfele liftezni. Ahhoz, hogy egynél nagyobb számokat is kiírjon a gép, természetesen meg kell szorozni valamennyivel. És amennyivel megszorozzuk, az a szám lesz a felső határa az RND által kiírt számok tartományának.

```
10 ? 6*RND(1)
20 GOTO 10
```

Ez a program 0 és 6 közötti számokat ír fel a képernyőre, tengernyi tizedessel. Ráadásul a 6-ot sohasem éri el, hiszen eredetileg mindig 1 alatt maradt.

Ha azt akarjuk, hogy 1 alatti számokat ne írjon (mert, mondjuk, egy dobókocka véletlenszerűen kipörgetett számait akarjuk utánozni), hozzá kell adnunk 1-et, így biztosan nem megy 1 alá, és így már eléri a 6-ot is. A tizedesektől pedig úgy szabadulhatunk meg, ha az INT utasítást használjuk, tehát közöljük a géppel, hogy csak az egész számokra vagyunk kíváncsiak.

A dobókockát utánzó programunk tehát így fog kinézni:

```
10 PRINT INT(6*RND(1))+1
20 GOTO 10
```

Ami persze a végtelenségig dobálja a kockát, tehát egy FOR...NEXT ciklusba ágyazva kell megfegyelmeznünk. Hány kockadobást akarunk? Sok társasjátékban a szabály általában kettőt vagy hármat engedélyez, tehát maradjunk a hárommal.

```
5 FOR I=1 TO 3
10 PRINT INT(6*RND(1))+1,
20 NEXT I
```

És kész a legegyszerűbb dobókocka-program, amely egymás mellé írja a kidobott három számot, és sem 1 alatti, sem 6 fölötti számot nem dob ki soha.

Most már azt is tudjuk, hogyan lehet a véletlenszerű érték határait beállítani. Ha 1 és 100 közötti egész számokat akarunk a géppel kiíratni, akkor ezt kell írni:

```
? INT(100*RND(1))+1
```

Ami persze nem jelenti azt, hogy az 50 és 100 közötti számokat így kellene kérni a géptől:

```
? INT(100*RND(1))+50
```

Miért is? Mert a szorzással beállítjuk ugyan a felső határt 100-ra, az összeadással viszont nemcsak az alsó határt emeljük meg 50-nel, hanem a felsőt is, tehát 50 és 150 közötti számokat csináltatunk a géppel. Mi tehát a teendő? Ez:

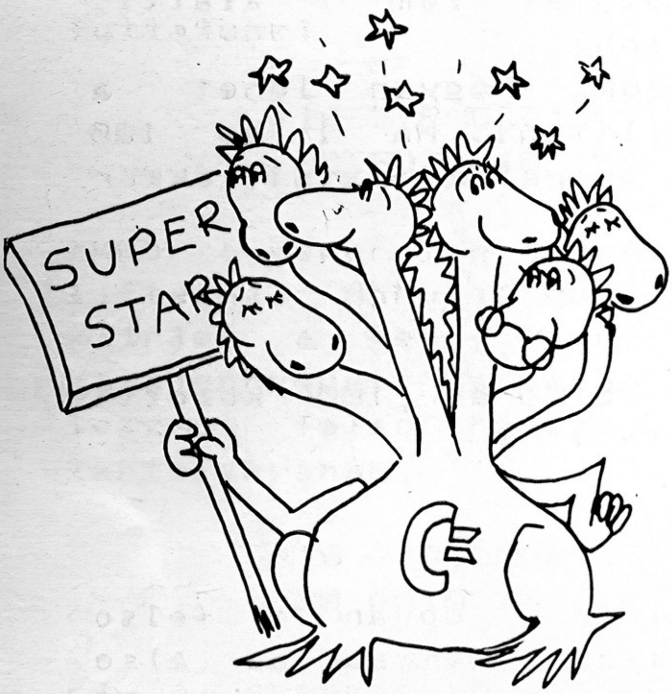
```
? INT(50*RND(1))+50
```

Világos? Akkor gépeljük be a következő kis programot, amely az esti égboltot rajzolja ki a képernyőre, ahogyan sorban kigyulladnak a csillagok.

```
10 REM CSILLAGOK
20 POKE53280,6:POKE53281,6
30 PRINT"♥":PRINT"■":REM CTRL+2
40 P=INT(39*RND(1))
50 Q=INT(22*RND(1))
60 FOR Y=0 TO Q:PRINT:NEXT Y
70 PRINT TAB(P)"*"
```

```
80 FOR N=1 TO(500*RND(1)):NEXT N
90 PRINT"▓":GOTO 40 : REM CLR/HOME
```

A programnak majdnem minden eleme ismerős, de van egy érdekesség a program 80-as sorában. Ez - mint látjuk - egy "fékező" sor, amely lassítja a program futását. Azt, hogy mennyivel lassítsa, egy véletlenszerű számmal határoztuk meg! Igy a csillagok kirajzolódásának gyorsasága (az adott határokon belül) teljesen szabálytalan.



Jó esetben összesen két magyarázattal tartozunk e programmal kapcsolatban. A 60-as sorban látunk egy PRINT utasítást, amelyhez nem írtuk oda, hogy mit írjon ki a gép. A PRINT után eddig mindig ott állt valamilyen változó, amely aztán megjelent a képernyőn. Nos, a PRINT, mint tudjuk, két dolgot csinál: egyrészt kiírja az utána következő változót (változókat) a képernyőre, másrészt - ha vesszővel vagy pontosvesszővel föl nem kérjük az ellenkezőjére - új sorba ugratja a cursort. Nos, itt épp ezt a tulajdonságát használjuk.

A PRINT utasítás, ha mögötte nem áll kiírandó változó, csak a második funkciót teljesíti, azaz egy sorral lejjebb viszi a cursort.

A 90-es sorban alaphelyzetbe visszük a cursort, utána pedig egy véletlenszerű számmal meghatározott számú sorral lejjebb ugratjuk. Igy a számítógép maga dönti el, hogy melyik sorba rajzolja a következő csillagot.

A másik ismeretlen kifejezés a TAB. Ez a "tabulátor" szó rövidítése - aki ismeri az írógépet, az tudja, hogy a táblázatok gépeléséhez használatos eszközről van szó (a latin "tabula", azaz "tábla" szóból ered).

A TAB utasításhoz szorosan hozzátartozik egy zárójeles szám, amely azt mondja meg a gépnek, hogy hány oszloppal mozdítsa el a cursort jobbra. Ez az utasítás:

```
? TAB(20)"*"
```

azt jelenti, hogy a képernyő 20. oszlopában ír ki egy csillagot a gép. (Vigyázat: a bal szélső oszlop száma 0!) A képernyőn 40 oszlop van, a TAB értéke azonban több is lehet 40-nél. TAB(40) a második sor első oszlopát jelenti, TAB(80) a harmadik sor első oszlopát, és így tovább. A legmagasabb érték, amit a TAB fölvehet, 255.

A programban ezt az oszlopszámot is egy véletlenszerű számmal határoztuk meg - így abszolúte semmit sem lehet tudni arról, hogy a gép hová fogja kirajzolni a következő csillagot. Az is lehet, hogy olyan helyre, ahol már van egy, tehát észre sem vesszük. Lehet fogadásokat kötni, hogy mennyi idő alatt teríti be a teljes képernyőt csillagokkal...

Pár nappal korábban megismerkedtünk a hol vidám, hol rosszkedvű babával. Most átalakítottuk szeszélyessé: teljesen szabálytalanul mosolyodik és komorodik el. Ki is színeztük egy kicsit; ha valaki el akar játszani vele, például átrajzolni az arcát, kicserélni a színeit, stb, akkor elég betölteni a kazettáról a korábbi változatot, és csak azokat a sorokat megváltoztatni, ahol különbséget látunk. (Ez persze majdnem minden sorra igaz, mert bevezettük az S$-t, amely 14 üres helyet tesz le minden sor elejére, hogy a babánk a képernyő közepére kerüljön.)

```
0 PRINT"■■■■":REM CRSR LE
3 POKE53280,3:POKE53281,1:PRINT"■"
5 S$="              ╗":REM CTRL+8
10 PRINTS$;"  ●●●●●●●●"
20 PRINTS$;" ●●●●●●●●●●"
30 PRINTS$;"●●●      ●●●"
40 PRINTS$;"●●        ●●"
50 PRINTS$;"●● ■ ▬|▬  ╗●●"
60 PRINTS$;"●●        ●●"
70 PRINTS$;"●●  ■O O  ╗●●":REM C=+7
80 PRINTS$;"●●   ■●   ╗●●":REM C=+3
90 PRINTS$;"●●        ●●"
100 PRINTS$;" ● ■╭─╮  ╗●":REM CTRL+1
110 PRINTS$;"  ■\ \  / / "
120 PRINTS$;"  ■ |╰─╯|  "
170 FORN=1TOINT(RND(1)*1500):NEXT
500 PRINT"■■■■■":REM CRSR LE
510 PRINTS$;"  ●●●●●●●●"
520 PRINTS$;" ●●●●●●●●●●"
530 PRINTS$;"●●●      ●●●"
540 PRINTS$;"●●        ●●"
550 PRINTS$;"●● ■▔▔ ▔▔ ╗●●"
```

```
560 PRINTS$;"●● ◘○  ○ ▜●●"
570 PRINTS$;"●● ◘◡  ◡ ▜●●"
580 PRINTS$;"●●   ▆●   ▜●●"
590 PRINTS$;"●● ■⁄   ` ▜●●"
600 PRINTS$;" ●  ■◡   ▜●"
610 PRINTS$;"   ■╲   ╱ "
620 PRINTS$;"   ■|◡|  "
630 PRINT"▒▒▒▒"
670 FORN=1TOINT(RND(1)*1000):NEXT
700 GOTO10
```

És itt a második kötet végén azok számára, akik nyomtatóhoz (printerhez) tudnak jutni, elmondjuk, hogyan lehet programunk listáját papírra nyomtatni, hogy ne csak a képernyőn szerkeszthessük, módosíthassuk.

Először is kössük össze a printert a számítógéppel, és kapcsoljuk be. (Ha nem kazettával, hanem mágneslemezzel dolgozunk, akkor a printert nem a számítógépbe kell csatlakoztatni, mert ott most a drive van, hanem a drive másik csatlakozójába.)

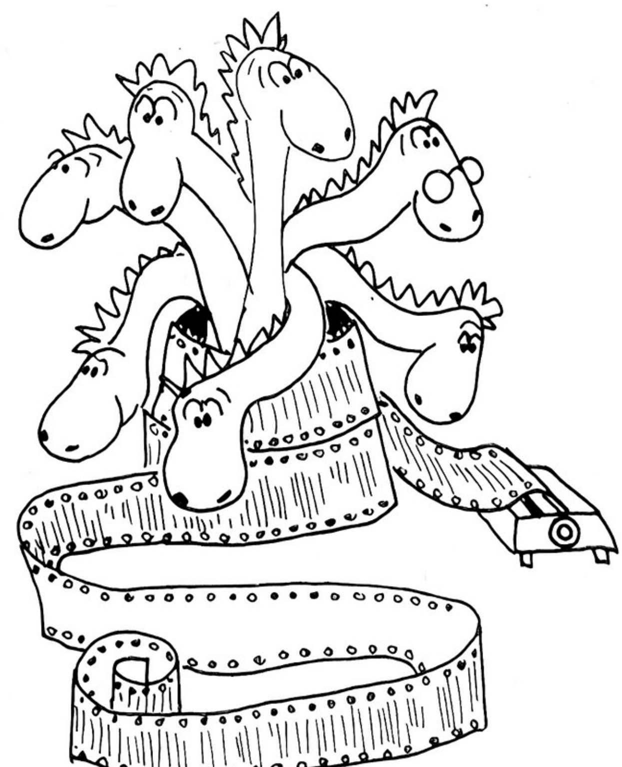
Állítsuk be a papírt a printeren. Utána töltsük be a gép memóriájába a programot, amelyet ki akarunk nyomtatni. Ezután gépeljük be a következő sorokat, mindegyik végén megnyomva a RETURNt:

```
OPEN 1,4
CMD 1
```

Ekkor a printer megelevenedik, és kiírja a papírra: READY.

Ekkor írjuk be a LIST utasítást, és nyomjuk meg a RETURNt!

A printer munkába lendül: nagy sebességgel kinyomtatja programunkat, majd, amikor elkészült vele, leáll.



Most gépeljük be a következő sort:

PRINT # 1

és ismét RETURN.

Vigyázat!!! Nagyon fontos! Ez a PRINT nem az a PRINT, amit eddig megismertünk! Ezt nem lehet rövidíteni kérdőjellel, hanem ki kell írni!

Az a kis kettős kereszt a 3-as billentyűn található meg, és SHIFTtel hívható elő.

Ha megnyomtuk a RETURNt, a printer nyekken egy picit, és ekkor beírhatjuk az utolsó üzenetet:

CLOSE 1

és újból RETURN.

Ha több programlistát akarunk egyszerre kinyomatni, akkor érdemes egy kis programot írni erre, a fenti utasításokból.

MÉG HÁTRAVAN EGY KÖTET; AMIT MOST HIÁNYOLNÁNK, ABBAN TALÁN MEGTALÁLHATÓ.

TARTALOM

JEGYZETEK

JEGYZETEK

JEGYZETEK

66,– Ft